2 Difference equations and modularity

The goals of this chapter are:

- to illustrate modularity and to describe systems in a modular way;
- to translate problems from their representation as a verbal description into their representation as discrete-time mathematics (difference equations); and
- to start investigating the simplest second-order system, the second-simplest module for analyzing and designing systems.

The themes of this chapter are modularity and the **representation** of verbal descriptions as discrete-time mathematics. We illustrate these themes with two examples, money in a hypothetical MIT endowment fund and rabbits reproducing in a pen, setting up difference equations to represent them. The rabbit example, which introduces a new module for building and analyzing systems, is a frequent visitor to these chapters. In this chapter we begin to study how that module behaves. Before introducing the examples, we illustrate what modularity is and why it is useful.

2.1 Modularity: Making the input like the output

A common but alas non-modular way to formulate difference and differential equations uses boundary conditions. An example from population growth illustrates this formulation and how to improve it by making it modular. The example is the population of the United States. The US population grows at an annual rate of roughly 1%, according to the *World FactBook* [2], and the US population is roughly 300 million in 2007. What will be the US population be in 2077 if the growth rate remains constant at 1%?

Pause to try 1. What is the population equation and boundary condition representing this information?

The difference equation for the population in year n is

$$p[n] = (1+r)p[n-1] \qquad \text{(population equation)},$$

where $r = 0.01$ is the annual growth rate. The boundary condition is

$$p[2007] = 3 \times 10^8 \qquad \text{(boundary condition)}.$$

To find the population in 2077, solve this difference equation with boundary condition to find $p[2077]$.

Exercise 1. What is p[2077]? How could you have quickly approximated the answer?

You might wonder why, since no terms are subtracted, the population equation is called a difference equation. The reason is by analogy with differential equations, which tell you how to find $f(t)$ from $f(t-\Delta t)$, with Δt going to 0. Since the discrete-time population equation tells us how to find $f[n]$ from $f[n-1]$, it is called a difference equation and its solution is the subject of the calculus of finite differences. When the goal – here, the population – appears on the input side, the difference equation is also a **recurrence relation**. What recurrence has to do with it is the topic of an upcoming chapter; for now take it as pervasive jargon.

The mathematical formulation as a recurrence relation with boundary condition, while sufficient for finding $p[2077]$, is messy: The boundary condition is a different kind of object from the solution to a recurrence. This objection to clashing categories may seem philosophical – in the colloquial meaning of philosophical as irrelevant – but answering it helps us to understand and design systems. Here the system is the United States. The input to the system is one number, the initial population $p[2007]$; however, the output is a sequence of populations $p[2008], p[2009], \ldots$. In this formulation, the system's output cannot become the input to another system. Therefore we cannot design large systems by combining small, easy-to-understand systems. Nor we can we analyze large, hard-to-understand systems by breaking them into small systems.

Instead, we would like a modular formulation in which the input is the same kind of object as the output. Here is the US-population question reformulated along those lines: *If $x[n]$ people immigrate into the United states in year n, and the US population grows at 1% annually, what is the population in year n?* The input signal is the number of immigrants versus time, so it is a sequence like the output signal. Including the effect of immigration, the recurrence is

$$\underbrace{p[n]}_{\text{output}} = \underbrace{(1+r)p[n-1]}_{\text{reproduction}} + \underbrace{x[n]}_{\text{immigration}} .$$

The boundary condition is no longer separate from the equation! Instead it is part of the input signal. This modular formulation is not only elegant; it is also more general than is the formulation with boundary conditions, for we can recast the original question into this framework. The recasting involves finding an input signal – here the immigration versus time – that reproduces the effect of the boundary condition $p[2007] = 3 \times 10^8$.

Pause to try 2. What input signal reproduces the effect of the boundary condition?

The boundary condition can be reproduced with this immigration schedule (the input signal):

$$x[n] = \begin{cases} 3 \times 10^8 & \text{if } n = 2007; \\ 0 & \text{otherwise.} \end{cases}$$

This model imagines an empty United States into which 300 million people arrive in the year 2007. The people grow (in numbers!) at an annual rate of 1%, and we want to know $p[2077]$, the output signal (the population) in the year 2077.

The general formulation with an arbitrary input signal is harder to solve directly than is the familiar formulation using boundary conditions, which can be solved by tricks and guesses. For our input signal, the output signal is

$$p[n] = \begin{cases} 3 \cdot 10^8 \times 1.01^{n-2007} & \text{for } n \geqslant 2007; \\ 0 & \text{otherwise.} \end{cases}$$

Exercise 2. Check that this output signal satisfies the boundary condition and the population equation.

In later chapters you learn how to solve the formulation with an arbitrary input signal. Here we emphasize not the method of solution but the modular formulation where a system turns one signal into another signal. This modular description using signals and systems helps analyze complex problems and build complex systems.

To see how it helps, first imagine a world with two countries: Ireland and the United States. Suppose that people emigrate from Ireland to the United States, a reasonable model in the 1850's. Suppose also that the Irish population has an intrinsic 10 annual decline due to famines and that another 10% of the population emigrate annually to the United States. Ireland and the United States are two systems, with one system's output (Irish emigration) feeding into the other system's input (the United States's immigration). The modular description helps when programming simulations. Indeed, giant population-growth simulations are programmed in this object-oriented way. Each system is an object that knows how it behaves – what it outputs – when fed input signals. The user selects systems and specifies connections among them. Fluid-dynamics simulations use a similar approach by dividing the fluid into zillions of volume elements. Each element is a system, and energy, entropy, and momentum emigrate between neighboring elements.

Our one- or two-component population systems are simpler than fluid-dynamics simulations, the better to illustrate modularity. Using two examples, we next practice modular description and how to represent verbal descriptions as mathematics.

2.2 Endowment gift

The first example for representing descriptions as mathematics involves a hypothetical endowment gift to MIT. A donor gives $\varpi 10^7$ dollars to MIT to support projects proposed and chosen by MIT undergraduates! MIT would like to use this fund for a long time and draw $\varpi 0.5 \times 10^6$ every year for a so-called 5% drawdown. Assume that the money is placed in a reliable account earning 4% interest compounded annually. How long can MIT and its undergraduates draw on the fund before it dwindles to zero?

Never make a calculation until you know roughly what the answer will be! This maxim is recommended by John Wheeler, a brilliant physicist whose most famous student was MIT alum Richard Feynman [9]. We highly recommend Wheeler's maxim as a way to build intuition. So here are a few estimation questions to get the mental juices flowing. Start with the broadest distinction, whether a number is finite or infinite. This distinction suggests the following question:

Pause to try 3. Will the fund last forever?

Alas, the fund will not last forever. In the first year, the drawdown is slightly greater than the interest, so the endowment capital will dwindle slightly. As a result, the next year's interest will be smaller than the first year's interest. Since the drawdown stays the same at $500,000 annually (which is 5% of the initial amount), the capital will dwindle still more in later years, reducing the interest, leading to a greater reduction in interest, leading to a greater reduction in capital... Eventually the fund evaporates. Given that the lifetime is finite, roughly how long is it? Can your great-grandchildren use it?

Pause to try 4. Will the fund last longer than or shorter than 100 years?

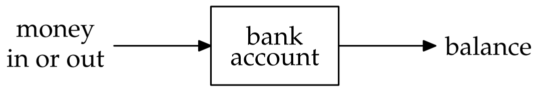
The figure of 100 years comes from the difference between the outflow – the annual drawdown of 5% of the gift – and the inflow produced by the interest rate of 4%. The difference between 5% and 4% annually is

$\delta = 0.01/\mathit{year}$. The dimensions of δ are inverse time, suggesting an endowment lifetime of $1/\delta$, which is 100 years. Indeed, if every year were like the first, the fund would last for 100 years. However, the inflow from interest decreases as the capital decreases, so the gap between outflow and inflow increases. Thus this $1/\delta$ method, based on extrapolating the first year's change to every year, overestimates the lifetime.

Having warmed up with two estimates, let's describe the system mathematically and solve for the true lifetime. In doing so, we have to decide what is the input signal, what is the output signal, and what is the system. The system is the least tricky part: It is the bank account paying 4 interest. The gift of $10 million is most likely part of the input signal.

Pause to try 5. Is the $500,000 annual drawdown part of the output or the input signal?

The drawdown flows out of the account, and the account is the system, so perhaps the drawdown is part of the output signal. No!! The output signal is what the system does, which is to produce or at least to compute a balance. The input signal is what you do to the system. Here, you move money in or out of the system:



The initial endowment is a one-time positive input signal, and the annual drawdown is a recurring negative input signal. To find how long the endowment lasts, find when the output signal crosses below zero. These issues of representation are helpful to figure out *before* setting up mathematics. Otherwise with great effort you create irrelevant equations, whereupon no amount of computing power can help you.

Now let's represent the description mathematically. First represent the input signal. To minimize the large numbers and dollar signs, measure money in units of $500,000. This choice makes the input signal dimensionless:

$$X = 20, -1, -1, -1, -1, \ldots$$

We use the notation that a capital letter represents the entire signal, while a lowercase letter with an index represents one sample from the signal. For example, P is the sequence of populations and $p[n]$ is the population in year n.

The output signal is

$$Y = 20, ?, ?, ?, \ldots$$

Pause to try 6. Explain why $y[0] = 20$.

The problem is to fill in the question marks in the output signal and find when it falls below zero. The difference equation describing the system is

$$y[n] = (1 + r)y[n-1] + x[n],$$

where r is the annual interest rate (here, $r = 0.04$). This difference equation is a first-order equation because any output sample $y[n]$ depends on the one preceding sample $y[n-1]$. The system that the equation represents is said to be a first-order system. It is the simplest module for building and analyzing complex systems.

Exercise 3. Compare this equation to the one for estimating the US population in 2077.

Now we have formulated the endowment problem as a signal processed by a system to produce another signal – all hail modularity! – and represented this description mathematically. However, we do not yet know how to solve the mathematics for an arbitrary input signal X. But here we need to solve it only for the particular input signal

$$X = 20, -1, -1, -1, -1, \ldots.$$

With that input signal, the recurrence becomes

$$y[n] = \begin{cases} 1.04 \cdot y[n-1] - 1 & n > 0; \\ 20 & n = 0. \end{cases}$$

The $y[0] = 20$ reflects that the donor seeds the account with 20 units of money, which is the \$10,000,000 endowment. The -1 in the recurrence reflects that we draw 1 unit every year. Without the -1 term, the solution to the recurrence would be $y[n] \sim 1.04^n$, where the $\sim$ symbol means 'except for a constant'. The -1 means that simple exponential growth is not a solution. However, -1 is a constant so it may contribute only a constant to the solution. That reasoning is dubious but simple, so try it first. Using a bit of courage, here is a guess for the form of the solution:

$$y[n] = A \cdot 1.04^n + B \qquad \text{(guess)},$$

where A and B are constants to be determined. Before finding A and B, figure out the most important characteristic, their signs. So:

Pause to try 7. Assume that this form is correct. What are the signs of A and B?

Since the endowment eventually vanishes, the variable term $A \cdot 1.04^n$ must make a negative contribution; so $A < 0$. Since the initial output $y[0]$ is positive, B must overcome the negative contribution from A; so $B > 0$.

Pause to try 8. Find A and B.

Solving for two unknowns A and B requires two equations. Each equation will probably come from one condition. So match the guess to the known balances at two times. The times (values of n) that involve the least calculation are the extreme cases $n = 0$ and $n = 1$. Matching the guess to the behavior at $n = 0$ gives the first equation:

$$20 = A + B \qquad (n = 0 \text{ condition}).$$

To match the guess to the behavior at $n = 1$, first find $y[1]$. At $n = 1$, which is one year after the gift, 0.8 units of interest arrive from 4% of 20, and 1 unit leaves as the first drawdown. So

$$y[1] = 20 + 0.8 - 1 = 19.8.$$

Matching this value to the guess gives the second equation:

$$19.8 = 1.04A + B \qquad (n = 1 \text{ condition}).$$

Both conditions are satisfied when $A = -5$ and $B = 25$. As predicted, $A < 0$ and $B > 0$. With that solution the guess becomes

$$y[n] = 25 - 5 \times 1.04^n.$$

This solution has a strange behavior. After the balance drops below zero, the 1.04^n grows ever more rapidly so the balance becomes negative ever faster.

Exercise 4. Does that behavior of becoming negative more and more rapidly indicate an incorrect solution to the recurrence relation, or an incomplete mathematical translation of what happens in reality?

Exercise 5. The guess, with the given values for A and B, works for $n = 0$ and $n = 1$. (How do you know?) Show that it is also correct for $n > 1$.

Now we can answer the original question: When does $y[n]$ fall to zero? Answer: When $1.04^n > 5$, which happens at $n = 41.035\ldots$. So MIT can draw on the fund in years $1, 2, 3, \ldots, 41$, leaving loose change in the account for a large graduation party. The exact calculation is consistent with the argument that the lifetime be less than 100 years.

Exercise 6. How much loose change remains after MIT draws its last payment? Convert to real money!

2.3 Rabbits

The second system to represent mathematically is the fecundity of rabbits. The *Encyclopedia Britannica* (1981 edition) states this population-growth problem as follows [6]:

A certain man put a pair of rabbits in a place surrounded on all sides by a wall. How many pairs of rabbits can be produced from that pair in a year if it is supposed that every month each pair begs a new pair which from the second month on becomes productive?

That description is an English representation of the original Latin. We first represent the verbal description mathematically and then play with the equations to understand how the system behaves. It is the simplest system beyond the first-order systems like the endowment, so it is an important module for building and analyzing complex systems.

2.3.1 From words to recurrence

Before representing the system mathematically, we describe it modularly using signals and systems by finding a system, an input signal, and an output signal. It is usually easiest to begin by looking for the system since it is the active element. The phrase 'surrounding on all sides by a wall' indicates a candidate for a system. The system is the inside of the wall, which is where the rabbits reproduce, together with the rules under which rabbits reproduce.

Pause to try 9. What is the input signal?

An input to the system is placing rabbits into it or taking them from it. The input signal is the number of pairs that enter the system at month n, where the signal would be negative if rabbits emigrate from the system to seek out tastier grass or other rabbit friends.

Pause to try 10. What is the output signal?

Some pairs are placed into the system as children (the immigrants); other pairs are born in the system (the native born). The sum of these kinds of pairs is the output signal.

To describe the system mathematically, decompose it by type of rabbit:

1. **children**, who cannot reproduce but become adults in one month; and
2. **adults**, who reproduce that month and thereafter.

Let $c[n]$ be the number of child pairs at month n and $a[n]$ be the number of adult pairs at month n. These intermediate signals combine to make the output signal:

$$f[n] = a[n] + c[n] \qquad \text{(output signal)}.$$

Pause to try 11. What equation contains the rule that children become adults in one month?

Because children become adults in one month, and adults do not die, the pool of adults grows by the number of child pairs in the previous month:

$$a[n] = a[n-1] + c[n-1] \qquad \text{(growing-up equation)}.$$

The two terms on the right-hand side represent the two ways to be an adult:

1. You were an adult last month ($a[n-1]$), or
2. you were a child last month ($c[n-1]$) and grew up.

The next equation says that all adults, and only adults, reproduce to make new children:

$$c[n] = a[n-1].$$

However, this equation is not complete because immigration also contributes child pairs. The number of immigrant pairs at month n is the input signal $x[n]$. So the full story is:

$$c[n] = a[n-1] + x[n] \qquad \text{(child equation)}$$

Our goal is a recurrence for $f[n]$, the total number of pairs. So we eliminate the number of adult pairs $a[n]$ and the number of child pairs $c[n]$ in favor of $f[n]$. Do it in two steps. First, use the growing-up equation to replace $a[n-1]$ in the child equation with $a[n-2] + c[n-2]$. That substitution gives

$$c[n] = a[n-2] + c[n-2] + x[n].$$

Since $f[n] = c[n] + a[n]$, we can turn the left side into $f[n]$ by adding $a[n]$. The growing-up equation says that $a[n]$ is also $a[n-1] + c[n-1]$, so add those terms to the right side and pray for simplification. The result is

$$\underbrace{c[n] + a[n]}_{f[n]} = \underbrace{a[n-2] + c[n-2]}_{f[n-2]} + x[n] + \underbrace{a[n-1] + c[n-1]}_{f[n-1]}.$$

The left side is $f[n]$. The right side contains $a[n-2] + c[n-2]$, which is $f[n-2]$; and $a[n-1] + c[n-1]$, which is $f[n-1]$. So the sum of equations simplifies to

$$f[n] = f[n-1] + f[n-2] + x[n].$$

The Latin problem description is from Fibonacci's *Liber Abaci* [10], published in 1202, and this equation is the famous Fibonacci recurrence but with an input signal $x[n]$ instead of boundary conditions.

This mathematical representation clarifies one point that is not obvious in the verbal representation: The number of pairs of rabbits at month n depends on the number in months $n-1$ and $n-2$. Because of this dependence on two preceding samples, this difference equation is a second-order difference equation. Since all the coefficients are unity, it is the simplest equation of that category, and ideal as a second-order system to understand thoroughly. To build that understanding, we play with the system and see how it responds.

2.3.2 Trying the recurrence

To play with the system described by Fibonacci, we need to represent Fibonacci's boundary condition that one pair of child rabbits enter the walls only in month 0. The corresponding input signal is $X = 1, 0, 0, 0, \ldots$. Using that X, known as an **impulse** or a **unit sample**, the recurrence produces (leaving out terms that are zero):

$$\begin{aligned}
f[0] &= x[0] = 1,\\
f[1] &= f[0] = 1,\\
f[2] &= f[0] + f[1] = 2,\\
f[3] &= f[1] + f[2] = 3,\\
&\ldots
\end{aligned}$$

When you try a few more lines, you get the sequence: $F = 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$. When you tire of hand calculation, ask a computer to continue. Here is slow Python code to print f[0], f[1],. . .,f[19]:

```
def f(n):
  if n < 2: return 1
  return f(n-1) + f(n-2)
print [f(i) for i in range(20)]
```

Exercise 7. Write the corresponding Matlab or Octave code, then rewrite the code in one of the languages – Python, Matlab, or Octave – to be efficient.

Exercise 8. Write Matlab, Octave, or Python code to find $f[n]$ when the input signal is $1, 1, 1, \ldots$. What is $f[17]$?

2.3.3 Rate of growth

To solve the recurrence in **closed form** – meaning an explicit formula for $f[n]$ that does not depend on preceding samples – it is helpful to investigate its approximate growth. Even without sophisticated techniques to find the output signal, we can understand the growth in this case when the input signal is the impulse.

Pause to try 12. When the input signal is the impulse, how fast does $f[n]$ grow? Is it polynomial, logarithmic, or exponential?

From looking at the first few dozen values, it looks like the sequence grows quickly. The growth is almost certainly too rapid to be logarithmic and, almost as certain, too fast to be polynomial unless it is a high-degree polynomial. Exponential growth is the most likely candidate, meaning that an approximation for $f[n]$ is

$$f[n] \sim z^n$$

where z is a constant. To estimate z, play with the recurrence when $n > 0$, which is when the input signal is zero. The $f[n]$ are all positive and, since $f[n] = f[n-1] + f[n-2]$ when $n > 0$, the samples are increasing: $f[n] > f[n-1]$. This bound turns $f[n] = f[n-1] + f[n-2]$ into the inequality

$$f[n] < f[n-1] + f[n-1].$$

So $f[n] < 2f[n-1]$ or $f[n]/f[n-1] < 2$; therefore the upper bound on z is $z < 2$. This bound has a counterpart lower bound obtained by replacing $f[n-1]$ by $f[n-2]$ in the Fibonacci recurrence. That substitution turns $f[n] = f[n-1] + f[n-2]$ into

$$f[n] > f[n-2] + f[n-2].$$

The right side is $2f[n-2]$ so $f[n] > 2f[n-2]$. This bound leads to a lower bound: $z^2 > 2$ or $z > \sqrt{2}$. The range of possible z is then

$$\sqrt{2} < z < 2.$$

Let's check the bounds by experiment. Here is the sequence of ratios $f[n]/f[n-1]$ for $n = 1, 2, 3, \ldots$:

$$1.0, 2.0, 1.5, 1.666\ldots, 1.6, 1.625, 1.615\ldots, 1.619\ldots, 1.617\ldots$$

The ratios seem to oscillate around 1.618, which lies between the predicted bounds $\sqrt{2}$ and 2. In later chapters, using new mathematical representations, you learn how to find the closed from for $f[n]$. We have walked two steps in that direction by representing the system mathematically and by investigating how $f[n]$ grows.

Exercise 9. Use a more refined argument to improve the upper bound to $z < \sqrt{3}$.

Exercise 10. Does the number 1.618 look familiar?

Exercise 11. [Hard!] Consider the same system but with one rabbit pair emigrating into the system every month, not only in month 0. Compare the growth with Fibonacci's problem, where one pair emigrated in month 0 only. Is it now faster than exponential? If yes, how fast is it? If no, does the order of growth change from $z \approx 1.618$?